CLAIMS

1. A method for performing coverability analysis in software, comprising:
performing a static analysis of software under test (SUT) so as to identify a plurality of dominating blocks in the SUT;
formulating respective coverability tasks for the dominating blocks of the SUT;
generating rules regarding behavior of the SUT corresponding respectively to the coverability tasks;
for each of the rules, running a symbolic model checker to test a behavioral model of the SUT, so as to produce respective results for the rules; and
computing a coverability metric for the SUT responsive to the results and the coverability tasks.

2. A method according to claim 1, and comprising writing the SUT in a programming language adapted to define at least one of a group of elements comprising a software element and a hardware element.

3. A method according to claim 1, wherein performing the static analysis of the SUT comprises:
identifying a set of dominating blocks in the SUT; and
solving a subset cover problem on the set of dominating blocks so as to identify the plurality of dominating blocks.

4. A method according to claim 3, wherein the set of dominating blocks comprises a set of all dominating blocks in the SUT, and wherein the plurality of dominating blocks comprises fewer blocks than the set of all dominating blocks in the SUT.

5. A method according to claim 4, wherein running

the symbolic model checker comprises performing a number of executions of the symbolic model checker smaller than a total number of all the dominating blocks in the SUT.

6. A method according to claim 1, wherein formulating the respective coverability tasks for the dominating blocks of the SUT comprises formulating coverability tasks by at least one of a group of methods comprising manual formulation and automatic formulation.

7. A method according to claim 1, wherein generating the rules regarding behavior of the SUT comprises generating rules by at least one of a group of methods comprising manual generation and automatic generation.

8. A method according to claim 1, wherein running the symbolic model checker to test the behavioral model of the SUT comprises:

evaluating the respective results so as to determine the truth or falsity of the rule; and

generating a list of uncoverable elements responsive to the respective results.

9. A method according to claim 1, wherein generating the rules regarding behavior of the SUT corresponding respectively to the coverability tasks comprises instrumenting the SUT by adding one or more statements and one or more auxiliary variables thereto, so as to facilitate evaluation of the rules.

10. A method according to claim 9, wherein instrumenting the SUT comprises:

determining a plurality of basic blocks comprised in the SUT; and

for each basic block:

defining an auxiliary variable for the block;

initializing the auxiliary variable to zero; and assigning the auxiliary variable a non-zero value upon execution of the basic block.

11. A method according to claim 9, wherein instrumenting the SUT comprises:

determining a plurality of basic blocks comprised in the SUT;

defining a single auxiliary variable for the SUT;

initializing the single auxiliary variable to zero; and

assigning a unique non-zero value to the single auxiliary variable upon execution of each basic block.

12. A method according to claim 1, wherein computing the coverability metric comprises:

evaluating an attained coverability responsive to the respective results produced by running the symbolic model checker;

evaluating an unattained coverability responsive to the respective results produced by running the symbolic model checker;

performing a comparison between the attained coverability and the coverability tasks;

calculating the coverability metric responsive to the comparison; and

analyzing the behavioral model of the SUT with respect to the unattained coverability.

13. A method according to claim 1, and comprising analyzing a design of the SUT, responsive to the coverability metric, for at least one of a group of properties comprising dead code, unattainable states, uncoverable statements, uncoverable states, unattainable transitions, unattainable variable values, and unreachable conditions.

14. A method according to claim 1, and comprising

applying a testing strategy chosen from one of a group of strategies comprising excluding uncoverable elements from coverage measurements, setting coverage goals responsive to the coverability metric, and determining a criterion for stopping testing responsive to the coverability metric.

15. A method according to claim 14, wherein the uncoverable elements comprise one or more elements chosen from a group of elements comprising uncoverable statements, uncoverable states, unattainable transitions, unattainable variable values, and unreachable conditions.

16. A method according to claim 1, wherein formulating the respective coverability tasks for the dominating blocks of the SUT comprises:

identifying a coverage model for the SUT;

defining a coverability model for the SUT responsive to the coverage model; and

generating the respective coverability tasks responsive to the coverability model.

17. A method for performing coverability analysis in software, comprising:

formulating first and second coverability tasks for software under test (SUT);

generating a rule regarding behavior of the SUT corresponding to the first coverability task;

running a symbolic model checker comprising an inflator to test a behavioral model of the SUT responsive to the rule so as to produce an inflated result; and

evaluating the second coverability task responsive to the inflated result.

18. A method according to claim 17, wherein formulating the second coverability task comprises choosing a plurality of coverability tasks from a set of

all coverability tasks for the SUT, and wherein evaluating the second coverability task comprises evaluating the plurality.

19. A method according to claim 17, wherein generating the rule regarding behavior of the SUT comprises:

performing a static analysis of the SUT comprising:

identifying a set of dominating blocks in the SUT; and

solving a subset cover problem on the set of dominating blocks so as to produce a plurality of dominating blocks; and

selecting the first coverability task responsive to the plurality.

20. A method according to claim 19, wherein selecting the first coverability task comprises:

identifying a greatest-influence dominating block having a largest set of dominated blocks comprised in the plurality; and

selecting the first coverability task responsive to the greatest-influence dominating block.

21. A method according to claim 19, wherein the set of dominating blocks comprises a set of all dominating blocks in the SUT, and wherein the plurality of dominating blocks comprises fewer blocks than the number of all the dominating blocks.

22. A method according to claim 17, wherein running the symbolic model checker comprises performing a number of executions of the symbolic model checker, wherein the number of executions is smaller than a total number of coverability tasks for the SUT.

23. A method according to claim 17, and comprising writing the SUT in a programming language adapted to

define at least one of a group of elements comprising a software element and a hardware element.

24. A method according to claim 17, wherein formulating the first and second coverability tasks for the SUT comprises formulating the tasks by at least one of a group of methods comprising manual formulation and automatic formulation.

25. A method according to claim 17, wherein generating the rule regarding behavior of the SUT comprises generating the rule by at least one of a group of methods comprising manual generation and automatic generation.

26. A method according to claim 17, wherein running the symbolic model checker comprises evaluating the inflated result and determining the truth or falsity of the rule responsive to the evaluation.

27. A method according to claim 17, wherein generating the rule comprises instrumenting the SUT by adding one or more statements and one or more auxiliary variables thereto, so as to facilitate evaluation of the rule.

28. A method according to claim 27, wherein instrumenting the SUT comprises:

determining a plurality of basic blocks comprised in the SUT; and

for each basic block:

defining an auxiliary variable for the block;

initializing the auxiliary variable to zero; and

assigning the auxiliary variable a non-zero value upon execution of the basic block.

29. A method according to claim 27, wherein instrumenting the SUT comprises:

determining a plurality of basic blocks comprised in the SUT;

defining a single auxiliary variable for the SUT;

initializing the single auxiliary variable to zero; and

assigning a unique non-zero value to the single auxiliary variable upon execution of each basic block.

30. A method according to claim 17, wherein running the symbolic model checker comprises producing the inflated result regardless of the truth or falsity of the rule.

31. A method according to claim 17, wherein evaluating the second coverability task responsive to the inflated result, comprises:

evaluating an attained coverability responsive to the inflated result from running the symbolic model checker;

evaluating an unattained coverability responsive to the respective results produced by running the symbolic model checker;

comparing the attained coverability with a plurality of all coverability tasks for the SUT;

calculating a coverability metric responsive to the comparison; and

analyzing the behavioral model of the SUT with respect to the unattained coverability.

32. A method according to claim 31, and comprising analyzing a design of the SUT, responsive to the coverability metric, for at least one of a group of properties comprising dead code, unattainable states, uncoverable statements, uncoverable states, unattainable transitions, unattainable variable values, and unreachable conditions.

33. A method according to claim 31, and comprising applying a testing strategy chosen from one of a group of strategies comprising excluding uncoverable elements from coverage measurements, setting coverage goals responsive to the coverability metric, and determining a criterion for stopping testing responsive to the coverability metric.

34. A method according to claim 33, wherein the uncoverable elements comprise one or more elements chosen from a group of elements comprising uncoverable statements, uncoverable states, unattainable transitions, unattainable variable values, and unreachable conditions.

35. A method according to claim 17, wherein running the symbolic model checker comprises:

performing a plurality of executions of an inflator program so as to produce a plurality of inflated results; and

evaluating the second coverability task responsive to the plurality of inflated results.

36. A method according to claim 17, wherein formulating the first and second coverability tasks for the SUT comprises:

identifying a coverage model for the SUT;

defining a coverability model for the SUT responsive to the coverage model; and

generating the first and second coverability tasks responsive to the coverability model.

37. Apparatus for performing coverability analysis in software, comprising a computing system which is adapted to perform a static analysis of software under test (SUT) so as to identify a plurality of dominating blocks in the SUT, formulate respective coverability

tasks for the dominating blocks of the SUT, generate rules regarding behavior of the SUT corresponding respectively to the coverability tasks, run a symbolic model checker to test a behavioral model of the SUT for each of the rules so as to produce respective results for the rules, and compute a coverability metric for the SUT responsive to the results and the coverability tasks.

38. Apparatus for performing coverability analysis in software, comprising a computer system which is adapted to formulate first and second coverability tasks for software under test (SUT), generate a rule regarding behavior of the SUT corresponding to the first coverability task, run a symbolic model checker comprising an inflator to test a behavioral model of the SUT responsive to the rule so as to produce an inflated result, and evaluate the second coverability task responsive to the inflated result.

39. A computer software product for performing coverability analysis in software, comprising a computer-readable medium having computer program instructions recorded therein, which instructions, when read by a computer, cause the computer to perform a static analysis of software under test (SUT) so as to identify a plurality of dominating blocks in the SUT, formulate respective coverability tasks for the dominating blocks in the SUT, generate rules regarding behavior of the SUT corresponding respectively to the coverability tasks, run a symbolic model checker to test a behavioral model of the SUT for each rule so as to produce respective results for the rules, and compute a coverability metric responsive to the results and the coverability tasks.

40. A computer software product for performing

coverability analysis in software, comprising a computer-readable medium having computer program instructions recorded therein, which instructions, when read by a computer, cause the computer to formulate first and second coverability tasks for software under test (SUT), generate a rule regarding behavior of the SUT corresponding to the first coverability task, run a symbolic model checker comprising an inflator to test a behavioral model of the SUT responsive to the rule so as to produce an inflated result, and evaluate the second coverability task responsive to the inflated result.